AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 2. ellenőrző dolgozat. 2015.03.27. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. Egy internetes áruházban minden termék árát euróban tartják nyilván, ugyanakkor azt a vásárló által használt pénznemben kell megjeleníteni. **Tervezzen** egy olyan osztályt (*Price*), ami a tárolt árat a vásárló által használt pénznemre átváltva adja meg! Legyen az osztálynak 7p
   - olyan konstruktora, amivel beállítható az Euróban megadott ár;
   - olyan tagfüggvénye, amivel beállítható az Euróban megadott ár;
   - olyan tagfüggvénye, amivel beállítható a használt pénznem árfolyama;
   - olyan tagfüggvénye, amivel beállítható/lekérdezhető a pénznem szöveges jele (pl. EUR, HUF).
   - olyan tagfüggvénye, amivel az aktuális pénznemre átváltva kérdezhető le az ár;

   Az árfolyamot és a pénznem szöveges jelét megadó függvény minden már létező és a későbbiekben létrejövő objektumpéldányra legyen hatással! **Valósítsa** meg az osztályt C++ nyelven úgy, hogy ne lehessen az adattagokhoz kívülről hozzáférni! Az alapértelmezett pénznem EUR legyen! Szöveg tárolásához használja az *std::string* osztályt! Működjön az alábbi kódrészlet!

```
Price ar1(10), ar2(20);
std::cout << ar1.getprice();   // 10 -et ír ki
Price::setrate(305.2); Price::setcurrency("HUF");
std::cout << ar1.getprice();   // 3052 -t ír ki
std::cout << ar2.getprice();   // 6104 -et ír ki
```

2. Az előző feladatban megtervezett osztályhoz készítsen inserter operátort (<<), ami képes az aktuális pénznemre átváltva kiírni az árat egy ostream típusú objektumra! A kiírt érték után írja ki a pénznem három betűs jelét is! 2.5p
3. Megoldását gondolatban bontsa .h és .cpp fájlokra! Jelölje, hogy mi kerül a .cpp állományba! 0.5p

megoldás:

```
class Price {
    static std::string currency;
    static double rate;
    double price;
public:
    Price(double _price) :price(_price) {}
    void set(double _price) { price = _price; }
    static std::string getcurrency() { return currency; }
    static void setcurrency(const char *_currency) { currency = _currency; }
    static void setrate(double _rate) { rate = _rate; }
    double getprice() const { return price * rate; }
};
```

```
// .cpp-be:
std::string Price::currency = "EUR";
double Price::rate = 1;

std::ostream& operator<<(std::ostream& os, const Price& p) {
    os << p.getprice() << p.getcurrency();
    return os;
}
```

BBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2015.03.27. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. Egy programban az időpontokat Unix-idő szerint tároljuk, ami nem más, mint greenwichi idő szerint 1970. január 1 óta eltelt másodpercek száma. A helyi idő ebből egy eltolás hozzáadásával számítható. **Tervezzen** egy olyan osztályt (*Ido*), ami Unix-időt tárol (*time_t*) és képes az időzónának megfelelő helyi időt szintén másodpercben megadni! Legyen az osztálynak 7p
   - paraméter nélkül hívható konstruktora, ami Unix-időt 0-ra állítja;
   - olyan tagfüggvénye, amivel beállítható az Unix-idő;
   - olyan tagfüggvénye, amivel lekérdezhető a helyi idő;
   - olyan tagfüggvénye, amivel beállítható az időzónának megfelelő eltolás;
   - olyan tagfüggvénye, amivel beállítható/lekérdezhető az időzóna jele (pl. CET, GMT, EET).

   Az időzónának megfelelő eltolás és az időzóna szöveges jelét megadó függvény minden már létező és a későbbiekben létrejövő objektumpéldányra legyen hatással! **Valósítsa** meg az osztályt C++ nyelven úgy, hogy ne lehessen az adattagokhoz kívülről hozzáférni! Az alapértelmezett időzóna UTC legyen! Szöveg tárolásához használja az *std::string* osztályt! Működjön az alábbi kódrészlet!

```
time_t most = time(NULL);      // a pillanatnyi Unix időt adja
Ido t1, t2; t2.set(most);      // t2-taktualis időre állítjuk be
std::cout << t1.localtime();   // 0-t ír ki
Ido::setoffs(3600);            // eltolás: 1*60*60 sec
Ido::setzone ("CET");          // közép-európai idő
std::cout << t1.localtime();   // 3600-at ír ki
std::cout << t2.localtime();   // most + 3600-at ír ki
```

2. Az előző feladatban megtervezett osztályhoz készítsen inserter operátort (<<), ami képes a lokális időt másodpercben kiírni egy ostream típusú objektumra! A kiírt érték után írja ki az időzóna jelét is! 2.5p
3. Megoldását gondolatban bontsa .h és .cpp fájlokra! Jelölje, hogy mi kerül a .cpp állományba! 0.5p

megoldás:

```
class Ido {
    static std::string zone;
    static time_t offs;
    time_t tim;
public:
    Ido() :tim(0) {}
    void set(time_t _tim) { tim = _tim; }
    std::string getzone() const { return zone; }
    static void setzone(const char *_zone) { zone = _zone; }
    static void setoffs(time_t _offs) { offs = _offs; }
    time_t localtime() const { return tim + offs; }
};
```

```
// .cpp-be:
std::string Ido::zone = "UTC";
time_t Ido::offs = 0;

std::ostream& operator<<(std::ostream& os, const Ido& t) {
    os << t.localtime() << t.getzone();
    return os;
}
```